Arquitectura multicapa de búsqueda por imagen aplicado a bases de conocimiento colaborativas para uso en dispositivos móviles

Sebastián Dapoto, Federico Cristina,
Verónica Artola, Javier Giacomantone

Instituto de Investigación en Informática LIDI. Facultad de Informática. UNLP
{sdapoto, fcristina, vartola, jog}@lidi.info.unlp.edu.ar

Resumen. En la actualidad, existe un considerable número de aplicaciones que permiten realizar búsquedas en bases de datos cuyo criterio es especificado mediante una imagen, en lugar de la tradicional utilización de palabras clave. El presente trabajo expone los avances realizados en el desarrollo de una arquitectura de código abierto multicapa para realizar este tipo de búsquedas y la posterior visualización de resultados; con la ventaja de brindar la versatilidad suficiente para lograr la escalabilidad y adaptación según las necesidades específicas de cada escenario de utilización. A su vez se detallan las mejoras realizadas en la lógica de correspondencia entre imágenes a fin de lograr una mayor robustez en los resultados.

Palabras clave: búsqueda por imagen, búsqueda visual inversa, SURF

1 Introducción

Las aplicaciones de búsqueda por imagen [1][2][3] simplifican la tarea al usuario, quien sólo requiere tomar una fotografía del objeto de interés y a partir de ésta, realizar la búsqueda en cuestión. Para esto se determina un conjunto de características de la imagen de entrada y se compara con las características de las imágenes almacenadas en la base de datos.

En la mayoría de los casos, las soluciones implementadas requieren de dos aplicaciones independientes, pero relacionadas funcionalmente. Se cuenta con la aplicación cliente en el dispositivo móvil, la cual se encarga de capturar la imagen, realizar el procesamiento de extracción de características e interactuar con la aplicación remota. Esta última escucha peticiones de los clientes, determina la mejor

correspondencia con las imágenes en la base de datos en función de las características extraídas y envía la información detallada relacionada con la imagen seleccionada.

En este tipo de arquitecturas, la aplicación cliente presenta un alto costo computacional, debido a la complejidad en las técnicas de extracción de características y detección de objetos. Más allá del gran avance en cuanto a la evolución de los dispositivos móviles, este requerimiento hace que dicha solución sea prohibitiva para un gran número de éstos. Además, este tipo de aplicaciones móviles se encuentran desarrolladas únicamente para un pequeño conjunto de plataformas donde las mismas pueden ejecutarse.

Es por este motivo que el presente trabajo plantea una arquitectura alternativa, basada en una aplicación Web, centralizando en el servidor la tarea de extracción de características, y relevando al cliente de la misma. El único requisito para el dispositivo móvil cliente es, además de una cámara que permite realizar las capturas, contar con la funcionalidad de navegador Web.

De esta manera, no solo es posible ampliar el espectro de dispositivos a utilizar, sino que también se heredan otros beneficios inherentes a aplicaciones Web, tales como:

- Procesamiento: Las tareas más costosas computacionalmente son realizadas en el servidor, mientras que el cliente web requiere únicamente presentar una simple interfaz web para la utilización de la solución por parte del usuario.
- Portabilidad: Más allá de la plataforma utilizada (Android, Symbian, Blackberry, iOS, Windows Mobile), al ser una aplicación Web no se requieren aplicaciones específicas para cada uno de los casos.
- Actualizaciones: Dado que el portal de la aplicación para el cliente se encuentra centralizado en el servidor, no se requieren descargas o actualizaciones de software específicas para su utilización.

Este tipo de solución puede ser utilizada en distintos escenarios, tales como salas de exposición, museos y recorridos interactivos, ya sea mediante una red de área local o Internet. Los siguientes apartados presentan el detalle de la solución planteada, en conjunto con las pruebas, resultados y conclusiones correspondientes.

2 Estado del arte

Entre las técnicas de correspondencia de imágenes más utilizadas se destacan básicamente los algoritmos *Scale-invariant feature transform* (SIFT) [4] y *Speeded*

Up Robust Features (SURF) [5]. En términos generales SURF es el algoritmo que logra los mejores resultados ponderando calidad y tiempo de respuesta.

Dentro de las implementaciones existentes que utilizan estas tecnologías o similares, una de las más conocidas es Google Goggles [6], la cual se apoya en la extensa base de conocimiento con que cuenta Google a fin de obtener la información resultante respecto de la búsqueda en cuestión. La aplicación de todas maneras se encuentra limitada a dispositivos móviles Android o iOS únicamente.

Otro ejemplo de aplicación de esta tecnología es TinEye [7]. La funcionalidad de la misma puede interpretarse como un motor inverso de búsqueda por imágenes, en donde el resultado es una nómina de URL que también hacen referencia a la imagen buscada en la Web. En este caso estamos en presencia de una solución independiente de la plataforma, dado que la misma es una aplicación Web.

Como último ejemplo, RevIMG [8] permite especificar una imagen (o una región de interés dentro de ésta) y una categoría relacionada sobre la cual se desea realizar la búsqueda. El motor de búsqueda presenta resultados en función de las formas, colores y dimensiones de las imágenes.

En todos los casos mencionados, la base de conocimiento se encuentra controlada por la entidad que gestiona el motor de búsqueda. La solución propuesta justamente tiene por finalidad la gestión colaborativa de los contenidos por parte de los usuarios.

3 Arquitectura de la solución

Tal como se explicó previamente, el uso de esta arquitectura permite el desarrollo independiente de cada una de las partes que la componen, separando el procesamiento de las imágenes de la visualización de los resultados.

Para esto, se cuenta con una capa de bajo nivel que provee el motor descriptor de imágenes y de detección de objetos, y con una capa de alto nivel que hace uso de la aplicación en cuestión. Existe además una capa intermedia que permite integrar las dos capas mencionadas, permitiendo la abstracción necesaria entre las mismas.

La solución implementada en el presente trabajo comprende el uso de *OpenSURF* [9] para la capa inferior, *Java Native Interface* [10] para la intermedia y *Java Server Pages* [11] para la superior.

3.1 Capa inferior: Correspondencia de imágenes

Desde el punto de vista computacional, la correspondencia entre dos imágenes es uno de los aspectos más costosos en visión por computadora [12]. Un método posible para analizar la correspondencia se basa en detectar características específicas y generar un descriptor para cada imagen.

Existen varias soluciones a la problemática recientemente expuesta. En particular, la solución desarrollada en el presente trabajo se apoya en la librería OpenSURF, la cual es una implementación del algoritmo SURF. Este algoritmo permite extraer los puntos de interés de una imagen con la particularidad de ser invariante con respecto a la rotación y a la escala. El mismo logra mejor tiempo de respuesta con respecto a soluciones previamente propuestas, debido al uso de imágenes integrales.

3.2 Capa superior: Interfaz de usuario

Una de las principales finalidades de la solución adoptada es que pueda ser utilizada en la mayor variedad de dispositivos móviles existente. Es por este motivo que se determinó la utilización de un cliente web como medio de acceso para el usuario. De esta manera no se presentan restricciones en cuanto a la plataforma a utilizar o los requerimientos en cuanto a la capacidad de procesamiento del dispositivo.

La arquitectura para la capa superior consta de un contenedor web con soporte para Java Server Pages (JSP), la cual es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, entre otros. Dicha capa permite al usuario cargar la imagen como criterio de búsqueda y presenta además los resultados correspondientes.

3.3 Capa intermedia: Integración entre capas

Debido a aspectos principalmente de performance, la capa inferior se encuentra desarrollada en lenguaje C, mientras que la capa superior está basada en tecnología Java, con lo cual es necesario contar con una capa intermedia que permita integrar ambas partes. Para lograr esto, se utilizó Java Native Interface (JNI).

Java Native Interface es un framework que permite a un programa escrito en Java ejecutándose en una máquina virtual, interactuar con programas escritos en otros lenguajes, como por ejemplo el lenguaje C.

Las capas inferiores de la solución son compiladas como una librería, la cual es cargada y accedida desde la instancia web de la aplicación en el servidor. De manera

simplificada, el servlet de la capa superior invoca métodos específicos de la librería, los cuales se encargan a su vez de llamar a los métodos nativos de la capa inferior.

Una vez realizado el procesamiento correspondiente en la capa inferior y obtenidos los resultados del caso, los mismos son propagados de manera inversa al orden de invocación, nuevamente mediante la capa intermedia, y devolviendo a la capa superior los datos a visualizar en el cliente.

4 Implementación

4.1 Aplicación

La figura 1 muestra la secuencia de pasos que abarca la utilización general de la solución, aplicado a una base de conocimiento con datos de artistas. El portal de búsqueda presenta un diseño minimalista a fin de simplificar el uso de la herramienta, así como de requerir el menor procesamiento posible por parte del dispositivo móvil.

Figura 1. Secuencia general de uso. a) portal de acceso, b) selección de imagen, c) previsualización de imagen a buscar, d) resultado obtenido.

4.2 Interacción Cliente - Servidor

La figura 2 muestra el comportamiento de cada una de las partes de la solución. El cliente realiza la petición al servidor, enviando una imagen a través de una red de área local (LAN) o a través de Internet. Una vez recibida la petición, el servidor realiza el procesamiento correspondiente y devuelve el resultado obtenido, el cual contiene la información en cuestión en conjunto con una imagen referencial del objeto.

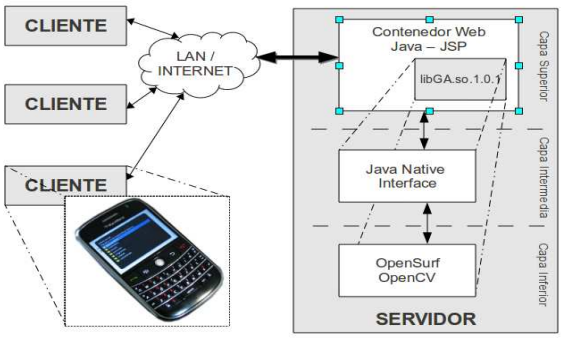

Figura 2. Arquitectura general de la solución

4.3 Optimización OpenSurf

La solución original mediante OpenSurf presentaba resultados aceptables incluso ante fotografías con poca luminosidad, movidas o fuera de foco. Sin embargo, se realizaron modificaciones adicionales a dicha solución a fin de lograr mayor robustez.

Básicamente se buscó refinar la calidad del resultado descartando puntos de correspondencia problemáticos, aplicando el siguiente criterio:

- Descarte por falsos positivos: si varios puntos de la imagen de entrada se corresponde con un solo punto de la imagen en la base de datos, dichos puntos son descartados del conjunto de puntos válidos.
- Descarte por orientación predominante: se ponderan los vectores direccionales a fin de determinar la dirección predominante de los puntos de correspondencia. Si alguno de estos puntos supera un umbral establecido, el mismo es eliminado del conjunto de puntos válidos.

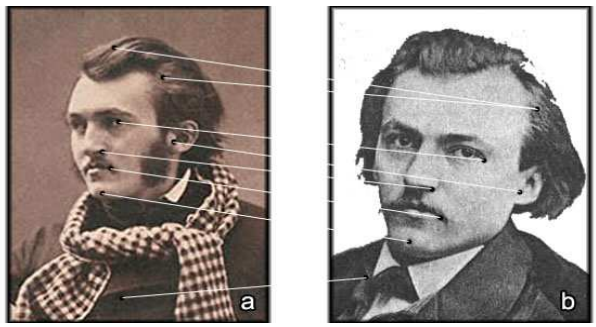

Figura 3. Esquema de la lógica de descarte. a) imagen de entrada cargada por el usuario, b) una de las imágenes de la base de datos.

Esta lógica se aplica en cada búsqueda, luego del procesamiento principal de OpenSurf. La figura 3 presenta un esquema simplificado del proceso. En la misma se observa que las dos correspondencias superiores son eliminadas por el criterio de falso positivo, mientras que la correspondencia extrema inferior es eliminada por el criterio de orientación predominante. De esta manera se logra depurar de las posibles correspondencias, las que eventualmente presentarán problemas al momento de determinar la imagen con mayor similitud.

5 Resultados

Para la realización de las pruebas se generó una base de conocimiento con un total de 50 fotografías de distintos libros. Éstas fueron tomadas con una resolución de 640x480 pixeles con el fin de contar con una cantidad razonable de puntos de interés, pero minimizando el tiempo de procesamiento al momento de determinar las correspondencias. Además, se verificó que a mayor resolución existe una mayor tasa de error en los resultados, dado que la cantidad de puntos de interés se ve incrementada, generando más posibilidades de correspondencias incorrectas.

Se realizaron varias series de pruebas, en las cuales las muestras fueron variando en lo que respecta a las siguientes características:

- Tamaño: desde 800x600 hasta 2592x1944 pixeles.
- Punto de vista: frontal, perspectiva lateral, perspectiva superior.
- Orientación: normal (vertical), apaisada (horizontal).
- Brillo/Contraste: normal, bajo, muy bajo.
- Foco: correcto, incorrecto (borroneado).

Las pruebas consistieron en comprobar los resultados utilizando OpenSurf y luego la solución optimizada. A fin de obtener un índice de calidad de los algoritmos, se tomó como principal dato de relevancia la cantidad de correspondencias válidas obtenidas entre la imagen de entrada y la seleccionada de la base de datos.

En el análisis de los resultados obtenidos se hizo especial énfasis en dos valores de referencia que representan la precisión y robustez del algoritmo para identificar la imagen correcta. La precisión indica en qué porcentaje de los casos se obtiene la imagen correcta. La robustez mide que tan lejos estuvo el algoritmo de devolver un resultado incorrecto. Este último valor fue calculado con el siguiente criterio:

- Si el resultado obtenido es correcto (el algoritmo determinó correctamente la imagen buscada), se compara con el caso más cercano en cuanto a la cantidad de correspondencias encontradas, de forma de medir cuán próximo estuvo a equivocarse.
- Si el resultado obtenido es incorrecto (la imagen resultante difiere de la buscada), se compara con la cantidad de correspondencias encontradas del resultado que se esperaba, de forma de medir cuán lejos estuvo de ser correcto.

La tabla 1 presenta un resumen de la serie de pruebas realizadas con el algoritmo original y optimizado, en las que se destacan los criterios de precisión y robustez previamente definidos.

En la figura 4 se puede observar una comparación entre los resultados logrados por ambos algoritmos para cada una de las muestras. En la parte izquierda de la figura se observa la comparación del valor de robustez. Se puede apreciar que el algoritmo optimizado obtiene valores ampliamente superiores en todos los casos, logrando excelentes resultados aun en las muestras más complejas. En la parte derecha de la figura se observa la comparación del valor de precisión, en la que se puede apreciar que el algoritmo optimizado logra encontrar la imagen buscada en todos los casos evaluados.

Tabla 1. Resumen de pruebas con ambos algoritmos

		ALGORITMO ORIGINAL				ALGORITMO OPTIMIZADO			
CARACTERISTICAS DE LA MUESTRA	PUNTOS DE INTERES (prom.)	PREC. (prom.)	CORRESP. IMGAGEN CORRECTA (prom.)	CORRESP. IMAGEN + PROX. (prom.)	ROBUSTEZ (prom.)	PREC. (prom.)	CORRESP. IMGAGEN CORRECTA (prom.)	CORRESP. IMAGEN + PROX. (prom.)	ROBUSTEZ (prom.)
1) 2592x1944 Persp. sup-lateral Orient. vertical Brillo/contr. normal Foco normal	7154,8	73,9%	257,5	193,6	24,8%	100,0%	91,3	17,0	81,4%
2) 1600x1200 Persp. frontal Orient. vertical Brillo/contr. normal Foco borroneado	3510,1	81,8%	282,6	170,6	39,6%	100,0%	138,5	17,1	87,7%
3) 1280x1024 Persp. frontal Orient. vertical Brillo/contr. normal Foco normal	2896,7	95,7%	312,3	154,0	50,7%	100,0%	178,4	16,2	90,9%
4) 1280x1024 Persp. frontal Orient. vertical Brillo/contr. escaso Foco normal	2103,2	91,3%	274,6	124,2	54,8%	100,0%	171,9	13,9	91,9%
5) 1280x1024 Persp. frontal Orient. horizontal Brillo/contr. normal Foco normal	2887,0	91,3%	293,8	149,2	49,2%	100,0%	105,3	16,1	84,7%
6) 800x600 Persp. sup-lateral Orient. vertical Brillo/contr. normal Foco normal	661,2	100,0%	104,1	23,1	57,8%	100,0%	67,3	7,0	89,5%

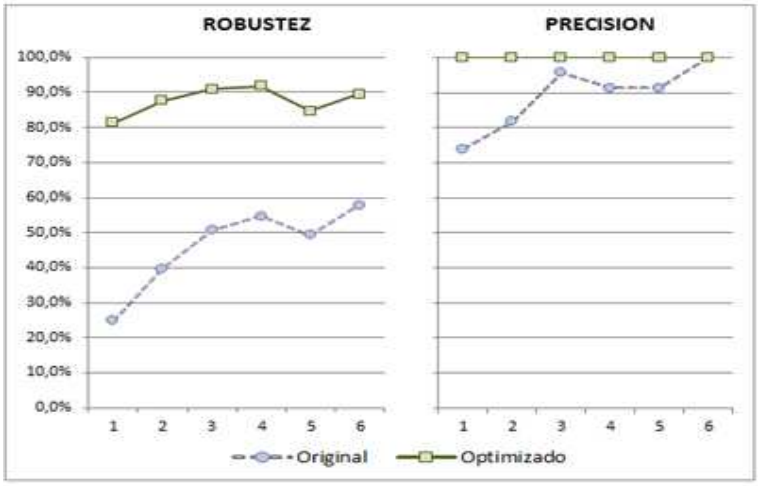

Figura 4. Comparación de robustez y precisión de los algoritmos.

6 Conclusiones

En el presente trabajo se detallaron los principales aspectos relacionados con una solución integral para el problema de búsqueda por imágenes y su acceso remoto mediante dispositivos móviles. La misma, al ser una solución Web, brinda los beneficios propios de este tipo de aplicaciones.

Por otra parte, al ser una arquitectura multicapa, es posible además modificar y/o ampliar de manera independiente cada una de estas capas en función de las necesidades específicas para cada caso.

Por último se especificaron las modificaciones realizadas al algoritmo de correspondencia de imágenes con el objetivo de lograr mayor precisión y robustez. Los resultados expuestos reflejan las mejoras logradas mediante estas modificaciones.

7 Agradecimientos

Este trabajo ha sido parcialmente financiado por el proyecto "D/031964/10" Formación de Recursos Humanos e Investigación en el Área de Visión por Computador e Informática Gráfica (FRIVIG) de la Agencia Española de Cooperación Inter-universitaria e Investigación Científica (AECID).

8 Referencias

1. Jonathan J. Hull, Xu Liu, Berna Erol, Jamey Graham, Jorge Moraleda: Mobile Image Recognition Architectures and Tradeoffs. HotMobile 2010 Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications (2010)
2. Boris Ruf, Effrosyni Kokiopoulou, Marcin Detyniecki: Mobile museum guide based on fast SIFT recognition. Lecture Notes in Computer Science, 2010, Volume 5811/2010, 170-183 (2010)
3. Nigel Davies, Keith Cheverst, Alan Dix, Andre Hesse: Understanding the Role of Image Recognition in Mobile Tour Guides. MobileHCI 2005 Proceedings of the 7th international conference on Human computer interaction with mobile devices & services (2005)
4. David G. Lowe: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60, 2 (2004), pp. 91-110
5. Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool: SURF: Speeded Up Robust Features. Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346--359, (2008)
6. Google Goggles, use pictures taken with your mobile phone to search the web. http://www.google.com/mobile/goggles/
7. TinEye, reverse image search engine. http://www.tineye.com/
8. RevIMG, reverse visual search engine. http://www.revimg.net/
9. Christopher Evans: Notes on the OpenSURF Library (2009). http://www.chrisevansdev.com/computer-vision-opensurf.html
10. JNI - The Java™ Native Interface - Java Developers. http://java.sun.com/docs/books/jni/download/jni.pdf
11. Java Server Pages. http://oracle.com/technetwork/java/javaee/jsp/index.html
12. Tomohiro Nakai, Koichi Kise, and Masakazu Iwamura: Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval. Lecture Notes in Computer Science, 2006, Volume 3872/2006, 541-552 (2006)